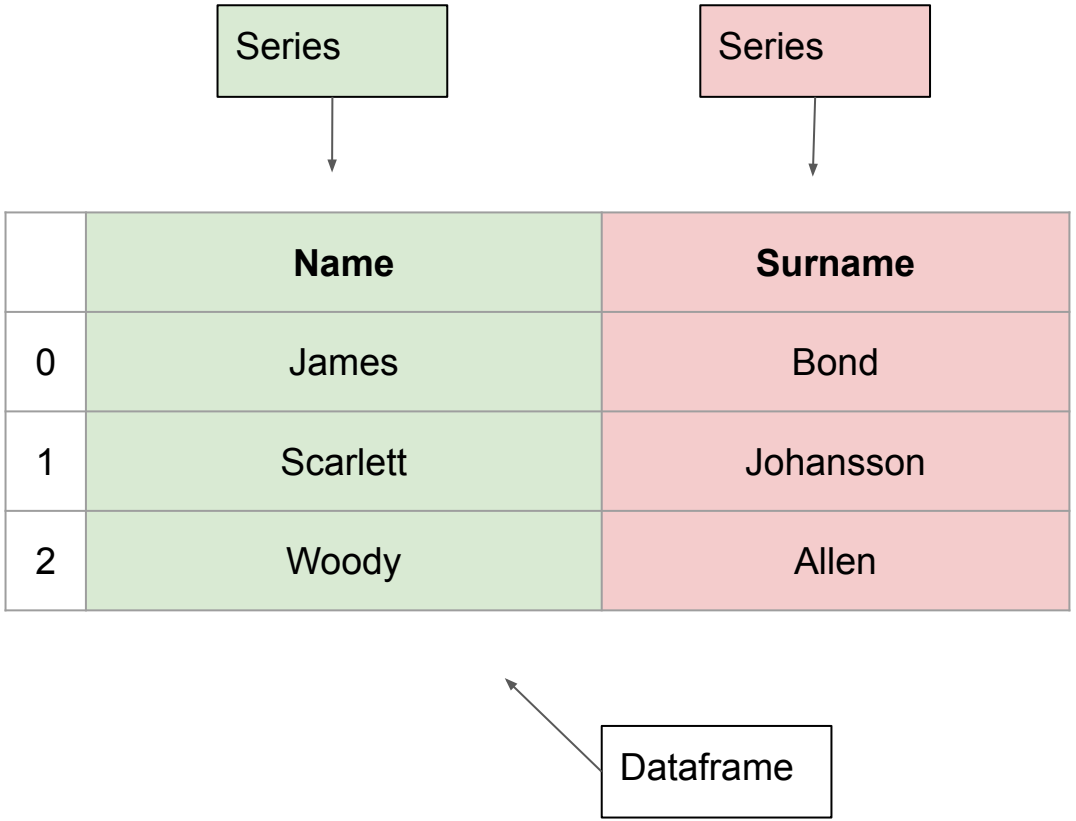
Введение в Pandas

Эдгар Казиахмедов

Что такое pandas?

- программная библиотека на языке Python для обработки и анализа данных;
- работа pandas с данными строится поверх библиотеки NumPy;
- Библиотека предоставляет *удобные* структуры данных и операции для работы с числовыми таблицами и временными рядами;
- Основными структурами данных являются **Series** и **DataFrame** (на самом деле еще **Panel**, но он уже не используется)

Pandas. Структуры данных



Pandas. Series. Конструктор класса

```
class pandas.Series(data=None, index=None, dtype=None,
                    name=None, copy=False, fastpath=False)
```

`data` - массив, итератор, словарь или скаляр
`index` - массив (по умолчанию это просто range)
`dtype` - тип данных
`name` - название объекта
`copy` - копирование данных
`fastpath` - никто не знает что это...

Pandas. Series. Основные методы

Создание Series из numpy массива

```
>>> import pandas as pd
>>> import numpy as np
>>> arr = np.array([“M”, “S”, “U”])
>>> ser = pd.Series(arr)
>>> ser
0    M
1    S
2    U
dtype: object
```

Pandas. Series. Основные методы

Создание Series из обычного листа

```
>>> import pandas as pd
>>> arr = [“M”, “S”, “U”]
>>> ser = pd.Series(arr)
>>> ser
0    M
1    S
2    U
dtype: object
```

Pandas. Series. Основные методы

Создание Series из листа значений и индексов

```
>>> import pandas as pd
>>> arr = [“M”, “S”, “U”]
>>> ser = pd.Series([1, 2, 3], arr)
>>> ser
M    1
S    2
U    3
dtype: int64
```

Pandas. Series. Основные методы

Создание Series из словаря

```
>>> import pandas as pd
>>> curr = {"rub" : 100, "usd" :1000, "euro" : 10}
>>> ser = pd.Series(curr)
>>> ser
rub      100
usd     1000
euro      10
dtype: int64
```

Pandas. Series. Основные методы

Создание Series из константы

```
>>> import pandas as pd
>>> curr = {"rub" : 100, "usd" :1000, "euro" : 10}
>>> ser = pd.Series(1, curr)
>>> ser
rub      1
usd      1
euro     1
dtype: int64
```

Pandas. Series. Основные методы

Test.csv:

```
Team,Name
1,Richard Hendricks
1,Jared Dunn
2,Gilfoyle
3,Dinesh
3,Gavin Belson
```

Создание Series из csv файла

```
>>> import pandas as pd
>>> ser = pd.Series.from_csv("test.csv")
>>> ser (ser.head(2))
Team                    Name
1         Richard Hendricks
1                Jared Dunn
2                  Gilfoyle
3                    Dinesh
3               Gavin Belson
dtype: object
```

Pandas. Series. Индексация

Индексация по порядковому номеру

```
>>> import pandas as pd
>>> ser = pd.Series.from_csv("test.csv")
>>> ser[0]
' Name'
>>> ser[1]
' Richard Hendricks'
>>> ser[0:2]
Team                    Name
1         Richard Hendricks
dtype: object
```

Pandas. Series. Индексация

Условия в индексации

```
>>> import pandas as pd
>>> ser = pd.Series.from_csv("test.csv")
>>> ser[ser.index < ‘2’]
1     Richard Hendricks
1            Jared Dunn
dtype: object
>>> ser[ser == "Richard Hendricks"]
1    Richard Hendricks
dtype: object
```

Pandas. Series. Индексация

Индексация по ключу (loc) и позиции (iloc)

```
>>> import pandas as pd
>>> ser = pd.Series.from_csv("test.csv")
>>> ser.loc['1':'2']
1    Richard Hendricks
1          Jared Dunn
2            Gilfoyle
dtype: object
>>> ser.iloc[0:2]
Team                  Name
1       Richard Hendricks
dtype: object
```

Pandas. Series. Операции

Сложение двух последовательностей (вычитание и т.д.)

```
>>> import pandas as pd
>>> a = pd.Series([5, 2, 3, 7], index=['a', 'b', 'c', 'd'])
>>> b = pd.Series([1, 6, 4, 9], index=['a', 'b', 'd', 'e'])
>>> a.add(b, fill_value=0) (без fill_value => NaN)
a     6.0
b     8.0
c     3.0
d    11.0
e     9.0
dtype: float64
```

Pandas. Series. Статистика

Среднее значение значений, наиб., наим. и т.д.

```
>>> import pandas as pd
>>> a = pd.Series([5, 2, 3, 7], index=['a', 'b', 'c', 'd'])
>>> a.mean()
4.25
>>> a.std(), a.max(), a.min()
(2.217355782608345, 7, 2)
>>> a.count()
4
>>> a.median()
4.0
```

Pandas. Series. Статистика

Общее описание

```
>>> import pandas as pd
>>> a = pd.Series([5, 2, 3, 7], index=['a', 'b', 'c', 'd'])
>>> a.describe()
count    4.000000
mean     4.250000
std      2.217356
min      2.000000
25%      2.750000
50%      4.000000
75%      5.500000
max      7.000000
```

Pandas. Series. Дополнительные операции

```
>>> import pandas as pd
>>> a = pd.Series([5, 2, 3, 7], index=['a', 'b', 'c', 'd'])
>>> b = pd.Series([1, 6, 4, 9], index=['a', 'b', 'd', 'e'])
>>> a.append(b)
>>> a.drop('a')
>>> a.dropna()
>>> a.loc['a'] = 2
>>> a.apply((lambda a: a + 2))
>>> a.astype("int32")
>>> a.where(a < 3, 1)
>>> a.to_csv("test2.csv")
```


Вопросы по
pandas.Series

Pandas. Dataframe. Конструктор класса

```
class pandas.DataFrame(data=None, index=None, columns=None,
                       dtype=None, name=None, copy=False)
```

data - 2-ый массив, итератор, словарь или скаляр
index - массив (по умолчанию это просто range)
columns - массив (по умолчанию это просто range)
dtype - тип данных
name - название объекта
copy - копирование данных

Pandas. DataFrame. Основные методы

Создание DataFrame из csv файла

```
>>> import pandas as pd
>>> df = pd.DataFrame.from_csv("test.csv")
>>> df (df.head(2))
      col1  col2
row1     1     2
row2     3     4
```

Test.csv:

```
,col1,col2
row1,1,2
row2,3,4
```

Pandas. DataFrame. Основные методы

Создание DataFrame из массива

```
>>> import pandas as pd
>>> import numpy as np
>>> data = np.array([['','col1','col2'],
                     ['row1',1,2], ['row2',3,4]])
>>> df = pd.DataFrame(data=data[1:,1:], index=data[1:,0],
                      columns=data[0,1:]))
>>> df
      col1  col2
row1     1     2
row2     3     4
```

Pandas. DataFrame. Основные методы

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> import pandas as pd
>>> import numpy as np
>>> df = pd.DataFrame.from_csv("test.csv")
>>> print(df.dtypes)
month     object
income     int64
spent      int64
dtype: object
>>> print(df.index)
>>> print(df.columns)
>>> print(df.values)
```

Pandas. DataFrame. Основные методы

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df.describe()
             income       spent
count    12.000000   12.000000
mean   145.500000   80.250000
std    105.824469   53.689554
min     40.000000   20.000000
25%     86.750000   43.750000
50%    100.000000   74.000000
75%    162.500000  100.000000
max    400.000000  200.000000
```

Pandas. DataFrame. Основные методы

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df.sort_values("income", ascending=False)
   month  income  spent
5    Jun     400    100
4    May     300    200
1    Feb     200    150
2    Mar     150    100
3    Apr     120     45
0    Jan     100     50
6    Jul     100     90
7    Aug      99     90
...
```

Pandas. DataFrame. Основные методы

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df.sort_values([“income”, “spend”],
ascending=False)
   month  income  spent
5    Jun     400    100
4    May     300    200
1    Feb     200    150
2    Mar     150    100
3    Apr     120     45
6    Jul     100     90
0    Jan     100     50
7    Aug      99     90
...
```

Pandas. DataFrame. Индексация

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df.income
0     100
1     200
2     150
3     120
…
>>> df['income']
>>> df[0:3]
  month  income  spent
0   Jan     100     50
1   Feb     200    150
2   Mar     150    100
```

Pandas. DataFrame. Индексация

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df[['income', 'spent']]
    income  spent
0      100     50
1      200    150
2      150    100
3      120     45
…
>>> df.loc[3:7:2, ['income']]
   income
3     120
5     400
7      99
```

Pandas. DataFrame. Индексация

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df.loc[2, ['income']]
income    150
Name: 2, dtype: object
>>> df.iloc[0:3,[0,1]]
  month  income
0   Jan     100
1   Feb     200
2   Mar     150
>>> >>> df.iloc[:,[1]]
    income
0      100
1      200 ...
```

Pandas. DataFrame. Условия

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df[df.income > 100]
  month  income  spent
1   Feb     200    150
2   Mar     150    100
3   Apr     120     45
4   May     300    200
5   Jun     400    100
>>> df[df['month'].isin(['Feb', 'Mar','Apr'])]
  month  income  spent
1   Feb     200    150
2   Mar     150    100
3   Apr     120     45
```

Pandas. DataFrame. Замена

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df.loc[9, ['income']] = 100
>>> df.loc[9, ['income']]
income    100

>>> df.loc[9, ['income']] = np.nan
>>> df.loc[9, ['income']]
income    NaN

>>> df.loc[:, ['income']] = np.linspace(0,
1000, len(df))
>>> df['income']= np.linspace(0,1000, len(df))
```

Pandas. DataFrame. Замена

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df['spent'] = 0.1 * df['income']
>>> df.head()
  month  income  spent
0   Jan     100   10.0
1   Feb     200   20.0
2   Mar     150   15.0
3   Apr     120   12.0
4   May     300   30.0

>>> df.to_csv("test2.csv")
```

Pandas. DataFrame. Замена

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> cols = df.columns.tolist()
>>> a = np.arange(len(cols))
>>> np.random.shuffle(a)
>>> cols = np.array(cols)[a]
>>> df = df[cols]
>>> df
   income  spent month
0     100     50   Jan
1     200    150   Feb
2     150    100   Mar
...
```

Pandas. DataFrame. Добавление столбца

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> rev = (df['income'] +df['spent']).values
>>> df['revenue'] = rev

>>> df.insert(1, "revenue", rev, True)
>>> df

>>> df = df.assign(revenue = rev)
>>> df.head()
  month  income  spent  revenue
0   Jan     100     50      150
1   Feb     200    150      350
…
```

Pandas. DataFrame. Добавление строки

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df = df[df.month != 'Dec']
>>> df.append({'month' : 'Dec', 'income' :
100, 'spent' : 50}, ignore_index=True)
>>> df.tail(2)
   month  income  spent
10   Nov      70     20
11   Dec     100     50

>>> df = df.append(pd.Series(['Dec', 100,
50], index=df.columns), ignore_index=True)
```

(можно передать несколько Series)

Pandas. DataFrame. Удаление

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

Удаление столбца income

```
>>> df.drop('income', 1)
```

Удаление строки с пп 0

```
>>> df.drop(0, 0)
```

Удаление нескольких строк

```
>>> df.drop([0,3], 0)
```

Удаление нескольких столбцов

```
>>> df.drop(['income', 'month'], 1)
```

Pandas. DataFrame. Применение функции

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df['income'] =
df['income'].apply((lambda a: a * 10))

>>> df.head()
  month  income  spent
0   Jan    1000     50
1   Feb    2000    150
2   Mar    1500    100
3   Apr    1200     45
4   May    3000    200
```

Pandas. DataFrame. Применение функции

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df['revenue'] = df.apply((lambda row:
row['income'] + row['spent']), axis=1)

>>> df.head()
  month  income  spent  revenue
0   Jan     100     50      150
1   Feb     200    150      350
2   Mar     150    100      250
3   Apr     120     45      165
4   May     300    200      500
```

Pandas. DataFrame. Применение функции

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df.apply((lambda col: col**2 if
col.name=='income' else col))

>>> df.head()
  month  income  spent
0   Jan   10000     50
1   Feb   40000    150
2   Mar   22500    100
3   Apr   14400     45
4   May   90000    200
```

Вопросы по pandas.DataFrame

Pandas. DataFrame. Замена NaN

Test.csv:

```
,month,income,spent
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
```

```
>>> df.loc[0:3, ['income']] = np.nan
>>> df.income =
df.income.fillna(df.income.mean())

>>> df.head()
  month  income  spent
0   Jan   147.0     50
1   Feb   147.0    150
2   Mar   147.0    100
3   Apr   147.0     45
4   May   300.0    200
```

Pandas. DataFrame. One-hot encoding

```
>>> one_hot = pd.get_dummies(df.month)
>>> pd.concat([df, one_hot], axis=1)

>>> df.drop(['month'], axis=1)
>>> df.head()
```

	income	spent	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	Oct	Sep
0	400.0	50	0	0	0	0	1	0	0	0	0	0	0	0
1	400.0	150	0	0	0	1	0	0	0	0	0	0	0	0
2	400.0	100	0	0	0	0	0	0	0	1	0	0	0	0
3	400.0	45	1	0	0	0	0	0	0	0	0	0	0	0
4	300.0	200	0	0	0	0	0	0	0	0	1	0	0	0

…

Pandas. DataFrame. Группы

Test.csv:

```
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
12,Dec,40,20
13,Jan,2000,200
14,Mar,90,45
```

```
>>> df = pd.DataFrame.from_csv("test.csv")
>>> gr = df.groupby(['month'])

>>> gr.first().head()
       income  spent
month
Apr       120     45
Aug        99     90
Dec        90     40
Feb       200    150
Jan       100     50
```

Pandas. DataFrame. Группы

Test.csv:

```
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
12,Dec,40,20
13,Jan,2000,200
14,Mar,90,45
```

```
>>> df = pd.DataFrame.from_csv("test.csv")
>>> gr = df.groupby(['month'])

>>> gr.count().head()
       income  spent
month
Apr         1      1
Aug         1      1
Dec         2      2
Feb         1      1
Jan         2      2
```

Pandas. DataFrame. Группы

Test.csv:

```
0,Jan,100,50
1,Feb,200,150
2,Mar,150,100
3,Apr,120,45
4,May,300,200
5,Jun,400,100
6,Jul,100,90
7,Aug,99,90
8,Sep,77,58
9,Oct,40,20
10,Nov,70,20
11,Dec,90,40
12,Dec,40,20
13,Jan,2000,200
14,Mar,90,45
```

```
>>> df = pd.DataFrame.from_csv("test.csv")
>>> gr = df.groupby(['month'])

>>> gr.std().head()
              income       spent
month
Apr              NaN         NaN
Aug              NaN         NaN
Dec        35.355339   14.142136
Feb              NaN         NaN
Jan     1343.502884  106.066017
```

Pandas. Аналогия с SQL запросами

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

SQL

```sql
SELECT total_bill, tip, smoker, time
FROM tips
LIMIT 5;
```

```sql
SELECT * FROM (
  SELECT
    t.*,
    RANK() OVER(PARTITION BY sex ORDER BY tip)
AS rnk
  FROM tips t
  WHERE tip < 2
)
WHERE rnk < 3
ORDER BY sex, rnk;
```

Pandas

```
tips[['total_bill', 'tip', 'smoker',
'time']].head(5)
```

```
(tips[tips['tip'] < 2]
.assign(rnk_min=tips.groupby(['sex'])['tip']
              .rank(method='min'))
              .query('rnk_min < 3')
              .sort_values(['sex', 'rnk_min']))
```

Pandas. Практика

А теперь попробуем поработать с **Iris** датасетом!